Enriched Modal Logics

Aniello Murano

Università degli Studi di Napoli “Federico II”

Wien - November 7, 2013
Motivations

Is the system correct?
Motivations

Formal Verification:

- **System** $\rightarrow$ A mathematical model $M$
- **Desired Behavior** $\rightarrow$ A formal specification $\psi$
- **Correctness** $\rightarrow$ A formal technique
Motivations

Formal Verification:

- **System** → A mathematical model $M$
- **Desired Behavior** → A formal specification $\psi$
- **Correctness** → A formal technique

- Model Checking: Does $M$ satisfies $\psi$?

The system has the required behavior
Motivations

Formal Verification:

- System
- Desired Behavior
- Correctness

→ A mathematical model $M$
→ A formal specification $\psi$
→ A formal technique

◆ Model Checking: Does $M$ satisfies $\psi$?
◆ Satisfiability: Is there $M$ for $\psi$?

The system has the required behavior
A Basic Model: Kripke Structure

A system can be represented as a **Kripke Structure**: a labeled-state transition graph

\[ M = (AP, S, S_0, R, Lab) \]

- **AP** is a set of atomic propositions.
- **S** is a finite set of states.
- **S_0 \subseteq S** is the set of initial states.
- **R \subseteq S \times S** is a transition relation, total: \( \forall s \in S, \exists s' . R(s, s') \).
- **Lab : S \rightarrow 2^{AP}** labels states with propositions true in that states.

A path is a system run!
System Specification

- Modal and Temporal logic allow description of the temporal ordering of events
System Specification

- Modal and Temporal logic allow description of the temporal ordering of events

- Two main families of logics:
  - Linear-Time Logics (LTL)
    - Each moment in time has a unique possible future.
    - LTL expresses path properties based on the paths state labels.
    - Useful for hardware specification.
  - Branching-Time Logics (CTL, CTL*, and μ-CALCULUS)
    - Each moment in time may split into various possible future.
    - CTL* expresses state properties from which LTL-like properties are satisfied in an existential or universal way.
    - Useful for software specification.

Aniello Murano - Enriched Modal Logics
μ-calculus is a very expressive logic

- Can express several practical properties.
- Corresponds to alternating parity tree automata
- Important connections with MSO
- Strictly subsumes classical logics such as CTL, LTL, CTL*, ...
- Identifies powerful classes of Description Logics

Decision problems:
  - Model checking: UP ∩ co-UP
  - Satisfiability: ExpTime-complete
μ-calculus limitations

- Several important constructs cannot be easily translated to the μ-calculus:
  - Inverse Programs to travel relations in backward
  - Graded modalities to enable statements on a number of successors
  - Nominals as propositional variables true exactly in one state
\(\mu\)-calculus limitations

- Several important constructs cannot be easily translated to the \(\mu\)-calculus:
  - Inverse Programs to travel relations in backward
  - Graded modalities to enable statements on a number of successors
  - Nominals as propositional variables true exactly in one state

- Extensions of the \(\mu\)-calculus with these abilities induces families of enriched \(\mu\)-calculi.

- Similarly, we can define families of enriched temporal logics.
Outline of the talk
I part

✓ Motivations

☑ Fully enriched μ-calculus
Outline of the talk

I part

- Motivations
- Fully enriched \( \mu \)-calculus
- Families of enriched \( \mu \)-calculi
  - full graded \( \mu \)-calculus (with inverse programs and graded mod.)
  - hybrid graded \( \mu \)-calculus (with graded modalities and nominals)
  - full hybrid \( \mu \)-calculus (with inverse programs and nominals)
Outline of the talk

I part

✓ Motivations

☐ Fully enriched \( \mu \)-calculus

☐ Families of enriched \( \mu \)-calculi
  ◆ full graded \( \mu \)-calculus (with inverse programs and graded mod.)
  ◆ hybrid graded \( \mu \)-calculus (with graded modalities and nominals)
  ◆ full hybrid \( \mu \)-calculus (with inverse programs and nominals)

☐ Satisfiability of fully enriched \( \mu \)-calculus: Undecidable
Outline of the talk

I part

✓ Motivations

 Fully enriched $\mu$-calculus

 Families of enriched $\mu$-calculi
   full graded $\mu$-calculus (with inverse programs and graded mod.)
   hybrid graded $\mu$-calculus (with graded modalities and nominals)
   full hybrid $\mu$-calculus (with inverse programs and nominals)

 Satisfiability of fully enriched $\mu$-calculus: Undecidable

 Satisfiability of the other families we consider: ExpTime-complete
   Upper bound via Fully Enriched Automata (FEA).
   The upper bound holds also in case numbers are coded in binary
Outline of the talk
II part

- Graded Computation Tree Logic (GCTL)

- ExpTime solution of the satisfiability problem for graded numbers coded in unary/binary
Outline of the talk
II part

- Graded Computation Tree Logic (GCTL)

- ExpTime solution of the satisfiability problem for graded numbers coded in unary/binary

- Open questions on GCTL and its extensions:
  - $GCTL^*$, PGCTL/PGCTL*, etc..
Outline of the talk
II part

- Graded Computation Tree Logic (GCTL)

- ExpTime solution of the satisfiability problem for graded numbers
coded in unary/binary

- Open questions on GCTL and its extensions:
  - GCTL*, PGCTL/PGCTL*, etc..

- Some achievements in open system verification.
I part: Enriched $\mu$-calculi
Some known results

- Satisfiability for Fully enriched $\mu$-calculus is undecidable [Bonatti, Peron 2004]
Some known results

- Satisfiability for Fully enriched $\mu$-calculus is undecidable [Bonatti, Peron 2004]

- ExpTime-completeness of satisfiability for enriched $\mu$-calculi:
  - $\mu$-calculus with inverse programs [Vardi’98]
  - $\mu$-calculus with graded modalities [Kupferman, Sattler, Vardi’02]
  - full hybrid logic [Sattler, Vardi’01]
  - full graded logic in unary coding [Calvanese, De Giacomo, Lenzerini’01]
The fully enriched $\mu$-calculus

- The $\mu$-calculus is a propositional modal logic with least($\mu$) and greatest ($\nu$) fixpoint operators [Kozen 1983].

- The fully enriched $\mu$-calculus extends the $\mu$-calculus with
  
  - graded modalities: $\langle n, \alpha \rangle$ (atleast formulas) and $[n, \alpha]$ (alldbut formulas)
  
  - nominals propositions: Nominal set $\text{Nom}$
  
  - inverse programs: Use of both program sets $\text{Prog}$ and $\text{Prog}^{-}$
The fully enriched $\mu$-calculus (Syntax)

- Let $AP$, $Var$, $Prog$, and $Nom$ be sets of atomic proposition, propositional variables, atomic, programs and nominals.

- Syntax:
  \[
  \varphi ::= \text{true} \mid \text{false} \mid p \mid \neg p \mid y \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \langle n, \alpha \rangle \varphi \mid [n, \alpha] \varphi \mid \mu y. \varphi(y) \mid \nu y. \varphi(y)
  \]
  where $p \in AP \cup Nom$, $y \in Var$, $n \in N$, and $\alpha$ is a program or its converse.
The fully enriched $\mu$-calculus (Syntax)

- Let $AP$, $Var$, $Prog$, and $Nom$ be sets of atomic proposition, propositional variables, atomic, programs and nominals

- Syntax:
  
  \[
  \varphi ::= \text{true} \mid \text{false} \mid \neg p \mid y \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \langle n, \alpha \rangle \varphi \mid [n, \alpha] \varphi \mid \mu y. \varphi(y) \mid \nu y. \varphi(y) 
  \]

  where $p \in AP \cup Nom$, $y \in Var$, $n \in N$, and $\alpha$ is a program or its converse

- Fragments of the fully enriched $\mu$-calculus:
  - full graded $\mu$-calculus (without nominals)
  - hybrid graded $\mu$-calculus (without inverse programs)
  - full hybrid $\mu$-calculus (without graded modalities)
Semantics: The enriched model

- The semantics of the fully enriched \( \mu \)-calculus is given with respect to enriched Kripke structures

\[ K = (AP \cup \text{Nom}, W, W_0, R, \text{Lab}) \]
Semantics: The enriched model

- The semantics of the fully enriched μ-calculus is given with respect to enriched Kripke structures

\[ K = (\text{AP} \cup \text{Nom}, W, W_0, R, \text{Lab}) \]

- In particular, R and Lab are enriched as follows:
  - \( R : \text{Prog} \rightarrow 2^W \times W \) assigns to programs transitions relation over \( S \)
  - \( \text{Lab} : \text{AP} \cup \text{Nom} \rightarrow 2^W \) assigns to propositions and nominal sets of states, where those assigned to each nominal are singletons.
Semantics: The enriched model

- The semantics of the fully enriched μ-calculus is given with respect to enriched Kripke structures

\[ K = (\text{AP} \cup \text{Nom}, W, W_0, R, \text{Lab}) \]

- In particular, R and Lab are enriched as follows:
  - \( R : \text{Prog} \rightarrow 2^W \times W \) assigns to programs transitions relation over \( S \)
  - \( \text{Lab}: \text{AP} \cup \text{Nom} \rightarrow 2^W \) assigns to propositions and nominal sets of states, where those assigned to each nominal are singletons.

- Given a Kripke structure, atomic propositions and boolean connectivities are interpreted as usual:
  - \( K \) satisfies the nominal \( n \) at the starting state \( r \), since \( \text{Lab}(n) = \{s\} \)
  - \( K \) does not satisfy \( q \) at \( r \), but at \( s \).
Semantics

- For a Kripke structure, the new modalities are interpreted as follows.
- $\langle n, \alpha \rangle \varphi$ holds in $w$ if $\varphi$ holds at least in $n+1$ $\alpha$-successors of $w$.
- $[n, \alpha] \varphi$ holds in $w$ if $\varphi$ holds in all but at most $n$ $\alpha$-successors of $w$. 
Semantics

- For a Kripke structure, the new modalities are interpreted as follows.
  - $\langle n, \alpha \rangle \phi$ holds in $w$ if $\phi$ holds at least in $n+1$ $\alpha$-successors of $w$.
  - $[n, \alpha] \phi$ holds in $w$ if $\phi$ holds in all but at most $n$ $\alpha$-successors of $w$.

- In $r$, $\langle 1, b \rangle p$ holds

Diagram:

```
q --b--> h --b--> q
|        |        |
|        |        |
|        |        |
r  \----/  s

p --b--> h --b--> p
```

Aniello Murano - Enriched Modal Logics
Semantics

- For a Kripke structure, the new modalities are interpreted as follows.
- $\langle n, \alpha \rangle \varphi$ holds in $w$ if $\varphi$ holds at least in $n+1$ $\alpha$-successors of $w$.
- $[n, \alpha] \varphi$ holds in $w$ if $\varphi$ holds in all but at most $n$ $\alpha$-successors of $w$.

In $r$,

- $\langle 1, b \rangle p$ holds
- $[1, b] p$ does not hold.
For a Kripke structure, the new modalities are interpreted as follows.

- $\langle n, \alpha \rangle \varphi$ holds in $w$ if $\varphi$ holds at least in $n+1$ $\alpha$-successors of $w$.
- $[n, \alpha] \varphi$ holds in $w$ if $\varphi$ holds in all but at most $n$ $\alpha$-successors of $w$.

In $r$, $\langle 1, b \rangle p$ holds.
In $r$, $[1, b] p$ does not hold.
In $r$, $[2, b] p$ holds.
Semantics

- For a Kripke structure, the new modalities are interpreted as follows.
- \( \langle n, \alpha \rangle \phi \) holds in \( w \) if \( \phi \) holds at least in \( n+1 \) \( \alpha \)-successors of \( w \).
- \([n, \alpha] \phi \) holds in \( w \) if \( \phi \) holds in all but at most \( n \) \( \alpha \)-successors of \( w \).

\begin{itemize}
  
  \item In \( r \), \( \langle 1, b \rangle p \) holds
  \item In \( r \), \( [1, b] p \) does not hold.
  \item In \( r \), \( [2, b] p \) holds.
  \item In \( s \), \( \langle 0, b^- \rangle h \) holds
\end{itemize}
Semantics

- For a Kripke structure, the new modalities are interpreted as follows.
  - $\langle n, \alpha \rangle \phi$ holds in $w$ if $\phi$ holds at least in $n+1$ $\alpha$-successors of $w$.
  - $[n, \alpha] \phi$ holds in $w$ if $\phi$ holds in all but at most $n$ $\alpha$-successors of $w$.

![Diagram showing the interpretation of modalities with nodes q, h, p, and r, s, and edges labeled b and r.]

- In $r$, $\langle 1, b \rangle p$ holds
- In $r$, $[1, b] p$ does not hold.
- In $r$, $[2, b] p$ holds.
- In $s$, $\langle 0, b^- \rangle h$ holds.

- $\nu$ and $\mu$ are useful to express liveness and safety:
  - $AGp$: $p$ always true along all $\alpha$-paths is $\nu X. p \land [0, \alpha] X$
  - $EFp$: there exists an $\alpha$-path where $p$ eventually holds is $\mu X. p \lor \langle 0, \alpha \rangle X$

- Note that $\langle 0, \alpha \rangle \phi$ is $\langle \alpha \rangle \phi$ and $[0, \alpha] \phi$ is $[\alpha] \phi$
Structure properties

- In branching-time temporal logic, important model features to simplify decisions reasonings are:
  - **Finite-model property:**
    - Is there a finite model satisfying the formula
    - It is possible to use exhaustive (brute-force) methods!
  - **Tree-model property:**
    - Is there a tree-model shape satisfying the formula
    - It is possible to use tree automata!

- In enriched $\mu$-calculus we need **forest structures** as models
Forest structures

- A forest $F \subseteq N^+$ is a collection of trees:

- The elements of $F$ are nodes, the degree of $F$ is the maximum number of node's successors, and $0$, $1$, and $2$ are roots of $F$.

- The set $T = \{r \cdot x \mid x \in N^* \text{ and } r \cdot x \in F\}$ is the tree of $F$ rooted in $r$. 
A forest $F \subseteq N^+$ is a collection of trees:

The elements of $F$ are nodes, the degree of $F$ is the maximum number of node’s successors, and 0, 1, and 2 are roots of $F$.

The set $T = \{ r \cdot x \mid x \in N^* \text{ and } r \cdot x \in F \}$ is the tree of $F$ rooted in $r$.

A Kripke structure $K$ is a forest structure if it induces a forest:

- Nodes $W$ represent a forest and the relation $R$ is defined over nodes, where each pair of successive nodes is labeled with one atomic program or its converse.

Aniello Murano - Enriched Modal Logics
A forest $F \subseteq N^*$ is a collection of trees:

- The elements of $F$ are nodes, the degree of $F$ is the maximum number of node's successors, and 0, 1, and 2 are roots of $F$.
- The set $T = \{r \cdot x \mid x \in N^* \text{ and } r \cdot x \in F\}$ is the tree of $F$ rooted in $r$.
- A Kripke structure $K$ is a forest structure if it induces a forest:
  - Nodes $W$ represent a forest and the relation $R$ is defined over nodes, where each pair of successive nodes is labeled with one atomic program or its converse.
- A Kripke structure $K$ is a quasi forest structure if it becomes a forest structure after deleting all the edges entering a root of $W$. 
A forest $F \subseteq N^+$ is a collection of trees:

- The elements of $F$ are nodes, the degree of $F$ is the maximum number of node's successors, and 0, 1, and 2 are roots of $F$.
- The set $T = \{r \cdot x \mid x \in N^* \text{ and } r \cdot x \in F\}$ is the tree of $F$ rooted in $r$.

A Kripke structure $K$ is a forest structure if it induces a forest:

- Nodes $W$ represent a forest and the relation $R$ is defined over nodes, where each pair of successive nodes is labeled with one atomic program or its converse.

A Kripke structure $K$ is a quasi forest structure if it becomes a forest structure after deleting all the edges entering a root of $W$. 

Aniello Murano - Enriched Modal Logics
A forest $F \subseteq N^+$ is a collection of trees:

- The elements of $F$ are nodes, the degree of $F$ is the maximum number of node’s successors, and 0, 1, and 2 are roots of $F$.
- The set $T = \{ r \cdot x \mid x \in N^* \text{ and } r \cdot x \in F \}$ is the tree of $F$ rooted in $r$.
- A Kripke structure $K$ is a forest structure if it induces a forest:
  - Nodes $W$ represent a forest and the relation $R$ is defined over nodes, where each pair of successive nodes is labeled with one atomic program or its converse.
- A Kripke structure $K$ is a quasi forest structure if it becomes a forest structure after deleting all the edges entering a root of $W$.
- $K$ is a tree structure if $W$ consists of a single tree.
Forest and tree model property

- Given a sentence $\varphi$ of the full graded $\mu$-calculus with $m$ at least subsentences and counting up to $b$
Given a sentence $\varphi$ of the full graded $\mu$-calculus with $m$ at least subsentences and counting up to $b$, $\varphi$ is satisfiable if $\varphi$ has a tree model whose degree is at most $m \cdot (b+1)$. 
Forest and tree model property

- Given a sentence $\phi$ of the full graded $\mu$-calculus with $m$ at least subsentences and counting up to $b$.

  $\phi$ is satisfiable

  $\phi$ has a tree model whose degree is at most $m \cdot (b+1)$.

- The hybrid graded $\mu$-calculus does not enjoy the tree model property.
Forest and tree model property

- Given a sentence $\phi$ of the full graded $\mu$-calculus with $m$ at least subsentences and counting up to $b$
  
  $\phi$ is satisfiable
  
  $\phi$ has a tree model whose degree is at most $m \cdot (b+1)$.

- The hybrid graded $\mu$-calculus does not enjoy the tree model property.

- Given a sentence $\phi$ of the hybrid graded $\mu$-calculus with $k$ nominals, $m$ at least subsentences and counting up to $b$
  
  $\phi$ is satisfiable
  
  $\phi$ has a quasi forest model whose degree is at most $\max\{k+1, m \cdot (b+1)\}$

Aniello Murano - Enriched Modal Logics
Solving enriched mu-calculi

- We use an automata-theoretic approach.
- In modal $\mu$-calculus, we translate a formula to an alternating parity tree automaton and check for its emptiness.
  - The translation is polynomial
  - Checking for emptiness can be done in ExpTime
  - Satisfiability of $\mu$-calculus is solvable in ExpTime.
- For the enriched $\mu$-calculi, we need an enriched version of parity tree automata.
- Let us first recall alternating automata on infinite tree...
Nondeterministic (binary) tree automata: NTA

- A infinite (binary) tree is $t : \{0, 1\}^* \to \Sigma$

- A **path** is an infinite sequence of nodes starting at the root

- An **NTA** is a tuple $A = \langle Q, \Sigma, \delta, Q_0, F \rangle$
  - $\delta : Q \times \Sigma \to 2^{Q \times Q}$ is a tree transition relation
  - Runs are binary trees labeled with states accordingly to $\delta$
  - $F$ is an acceptance condition satisfied on each path of a run

- A $(\Sigma$-labeled) tree $t$
A run \( r : \{0,1\}^* \rightarrow Q \) is built in accordance with \( \delta \) and \( r(\varepsilon) \in Q_0 \).
Thus, runs are \( Q \)-labeled trees.

Let \((q,q) \in \delta(p,a)\) and \(q_0\) initial state

![Diagram](image-url)
A run \( r : \{0,1\}^* \rightarrow Q \) is built in accordance with \( \delta \) and \( r(\varepsilon) \in Q_0 \).

Thus, runs are \( Q \)-labeled trees.

Let \((q,q) \in \delta(p,a)\) and \(q_0\) initial state.
A run \( r : \{0,1\}^* \rightarrow Q \) is built in accordance with \( \delta \) and \( r(\varepsilon) \in Q_0 \). Thus, runs are \( Q \)-labeled trees.

Let \( (q,q) \in \delta(p,a) \) and \( q_0 \) initial state.

A run is accepting if the acceptance condition is satisfied on every path.
Alternating automata on infinite trees

- An alternating (finite-state) automaton on infinite $\Sigma$-labeled $D$-trees is a tuple

$$A = \langle Q, \Sigma, \delta, q_0, F \rangle$$

- $\delta : (Q \times \Sigma) \rightarrow B^+(D \times Q)$
- positive Boolean formulas of pairs of directions and states

For example

$$\delta(p,a) = (1,p) \land (1,q)$$

$\Sigma$-labeled binary tree
A run on a $\Sigma$-labeled $D$-trees is a $(D^* \times Q)$-labeled tree. The root is labeled with $(\epsilon, q_0)$ and labels of each node and its successors must satisfy the $\delta$.
A run on a $\Sigma$-labeled D-trees is a $(D^* \times Q)$-labeled tree. The root is labeled with $(\varepsilon, q_0)$ and labels of each node and its successors must satisfy the $\delta$

$\delta(q_0,a)=((0,q_1) \lor (0,q_2)) \land (0,q_3) \land (1,q_3)$

Let $S= \{(0,q_1), (0,q_3), (1,q_3)\}$. 

---

A binary tree $T$ 

The corresponding run $r$
A run on a $\Sigma$-labeled $D$-trees is a $(D^* \times Q)$-labeled tree. The root is labeled with $(\varepsilon, q_0)$ and labels of each node and its successors must satisfy the $\delta$

$\delta(q_0,a)=((0,q_1)\lor (0,q_2)) \land (0,q_3) \land (1,q_3)$

Let $S=\{(0,q_1), (0,q_3), (1,q_3)\}$.

There is no one-to-one correspondence between nodes of $T$ and $r$. 
A run on a \( \Sigma \)-labeled D-trees is a \((D^* \times Q)\)-labeled tree. The root is labeled with \((\varepsilon, q_0)\) and labels of each node and its successors must satisfy the \(\delta\):

\[
\delta(q_0, a) = \left( (0, q_1) \lor (0, q_2) \right) \land (0, q_3) \land (1, q_3)
\]

Let \(S = \{(0, q_1), (0, q_3), (1, q_3)\}\). There is no one-to-one correspondence between nodes of \(T\) and \(r\).

As in nondeterministic automata, a run is accepting if the acceptance condition is satisfied on every path.
Fully Enriched Automata

- Fully enriched automata (FEA) run on infinite labeled forests $\langle T,V \rangle$.
- FEA generalize alternating automata on infinite trees as the fully enriched $\mu$-calculus extends the standard $\mu$-calculus:
Fully Enriched Automata

- Fully enriched automata (FEA) run on infinite labeled forests \( \langle T, V \rangle \).
- FEA generalize alternating automata on infinite trees as the fully enriched \( \mu \)-calculus extends the standard \( \mu \)-calculus:
  - **Move up to a predecessor of a node**
    (by analogy with inverse programs)
  - **Move down to at least n or all but n successors**
    (by analogy with graded modalities)
  - **Jump directly to the roots of the input forest**
    (which are the analogues of nominals).
Fully Enriched Automata

- Fully enriched automata (FEA) run on infinite labeled forests $\langle T,V \rangle$.
- FEA generalize alternating automata on infinite trees as the fully enriched $\mu$-calculus extends the standard $\mu$-calculus:
  - Move up to a predecessor of a node (by analogy with inverse programs)
  - Move down to at least $n$ or all but $n$ successors (by analogy with graded modalities)
  - Jump directly to the roots of the input forest (which are the analogues of nominals).

- $\delta(q,\sigma)$ is a positive boolean combination of pairs of directions and states.

- Formally,
  - $\delta: Q \times \Sigma \to B^+(D_b \times Q)$, where $D_b$ can be $-1$, $\varepsilon$, $\langle \text{root} \rangle$, $[\text{root}]$, $\langle n \rangle$, or $[n]$, with $0 \leq n \leq b$.
  - $(-1, q)$ and $(\varepsilon, q)$ send a copy to the predecessor and to the current node.
  - $(\langle \text{root} \rangle, q)$ and $([\text{root}], q)$ send a copy to some or all roots of the forest.
  - $(\langle n \rangle, q)$ and $([n], q)$ send a copy in state $q$ to $n+1$ and all but $n$ successors of the current node, respectively.
Runs for FEA

- For a FEA $A$ with a transition $\delta: Q \times \Sigma \to B^+(D_b \times Q)$
- A run over a forest $\langle F,V \rangle$ is a $(F \times Q)$-labeled tree, built in accordance with $\delta$ and $r(\epsilon) = (c, q_0)$, for a root $c$ of $F$. 

Aniello Murano - Enriched Modal Logics
Runs for FEA

- For a FEA $A$ with a transition $\delta: Q \times \Sigma \rightarrow B^+ (D_b \times Q)$
- A run over a forest $\langle F, V \rangle$ is a $(F \times Q)$-labeled tree, built in accordance with $\delta$ and $r(\varepsilon) = (c, q_0)$, for a root $c$ of $F$. 

\[ \langle F, V \rangle: \]

\[0 \quad \quad \quad \quad 1 \]

\[00 \quad 01 \quad 02 \quad 10 \quad 11 \quad 12 \]

\[r: \]

\[c_1, q_0 \]

\[0 \quad 1 \]

\[00 \quad 01 \]

\[m \]
Runs for FEA

- For a FEA $A$ with a transition $\delta: Q \times \Sigma \rightarrow B^+(D_b \times Q)$
- A run over a forest $\langle F, V \rangle$ is a $(F \times Q)$-labeled tree, built in accordance with $\delta$ and $r(\varepsilon) = (c, q_0)$, for a root $c$ of $F$.
- Let $r(0) = (11, q)$, $V(11) = a$, and
  \[\delta(q, a) = (-1, q_1) \land ((\langle \text{root} \rangle, q_2) \lor ([\text{root}], q_3))\]
Runs for FEA

- For a FEA $A$ with a transition $\delta: Q \times \Sigma \rightarrow B^+(D_b \times Q)$
- A run over a forest $\langle F,V \rangle$ is a $(F \times Q)$-labeled tree, built in accordance with $\delta$ and $r(\epsilon) = (c, q_0)$, for a root $c$ of $F$.
- Let $r(0) = (11, q)$, $V(11) = a$, and
  
  $\delta(q, a) = (-1, q_1) \land ((\langle \text{root} \rangle, q_2) \lor ([\text{root}], q_3))$

- Let $S = \{(-1, q_1), (\langle \text{root} \rangle, q_2)\}$.

### Diagram

- $\langle F,V \rangle$: Diagram of a forest $\langle F,V \rangle$ with states 0, 1, and labels 00, 01, 02, 10, a, 12.
- $r$: Diagram of a run $r$ starting at $c_1, q_0$ and moving through states 11, q, 1, $c_2, q_2$, and $1, q_1$.
Runs for FEA

- For a FEA $A$ with a transition $\delta: Q \times \Sigma \rightarrow B^{+}(D_{b} \times Q)$
- A run over a forest $\langle F,V \rangle$ is a $(F \times Q)$-labeled tree, built in accordance with $\delta$ and $r(\varepsilon) = (c, q_{0})$, for a root $c$ of $F$.
- Let $r(0) = (11, q)$, $V(11) = a$, and
  \[ \delta(q,a) = (-1, q_{1}) \land ((<\text{root}>, q_{2}) \lor ([\text{root}], q_{3})) \]
- Let $S = \{(-1, q_{1}), (\langle \text{root} \rangle, q_{2})\}$.

- We use a parity condition.
Acceptance conditions

- **Büchi condition:** $F \subseteq Q$. A run $r$ is accepting iff for every path, there exists a final state appearing infinitely often.
- Formally, a run is accepting if for each path $\pi$, $\text{Inf}(r|\pi) \cap F \neq \emptyset$.
Acceptance conditions

- **Büchi condition**: \( F \subseteq Q \). A run \( r \) is accepting iff for every path, there exists a final state appearing infinitely often.

- Formally, a run is accepting if for each path \( \pi \), \( \text{Inf}(r|\pi) \cap F \neq \emptyset \)

- **Parity condition**: \( F = \{F_1, \ldots, F_m\} \). A run \( r \) is accepting if for each path \( \pi \) in \( r \) the minimal \( i \) for which \( \text{Inf}(r|\pi) \cap F \neq \emptyset \) is even.
Acceptance conditions

- **Büchi condition**: \( F \subseteq Q \). A run \( r \) is accepting iff for every path, there exists a final state appearing infinitely often.
- Formally, a run is accepting if for each path \( \pi \), \( \text{Inf}(r|\pi) \cap F \neq \emptyset \).

- **Parity condition**: \( F = \{F_1, \ldots, F_m\} \). A run \( r \) is accepting if for each path \( \pi \) in \( r \) the minimal \( i \) for which \( \text{Inf}(r|\pi) \cap F \neq \emptyset \) is even.

- **Emptiness**:
  - Nondeterministic Buchi Tree Automata (NBT) : PTime-Complete
  - Alternating Buchi Tree Automata (ABT) : ExpTime-Complete
  - Nondeterministic Parity Tree Automata (NPT) : UP \( \cap \) Co-UP
  - Alternating Parity Tree Automata (APT) : ExpTime-Complete
Solving the satisfiability problem

- We show that the satisfiability problem for enriched $\mu$-calculus formulas (except for fully enriched ones) is EXPTime-Complete
Solving the satisfiability problem

- We show that the satisfiability problem for enriched $\mu$-calculus formulas (except for fully enriched ones) is EXPTime-Complete
- Lower Bound: Satisfiability for the $\mu$-calculus is EXPTime-hard [Fisher Ladner 1979]
Solving the satisfiability problem

- We show that the satisfiability problem for enriched $\mu$-calculus formulas (except for fully enriched ones) is EXPTime-Complete
- Lower Bound: Satisfiability for the $\mu$-calculus is EXPTime-hard [Fisher Ladner 1979]
- Upper Bound: We use an automata-theoretic approach:
  - Given a sentence $\varphi$ of the full graded $\mu$-calculus that has $m$ at least sub-sentences and counts up to $b$, we can construct a FEA $A_\varphi$ that
    - accepts the set of tree models of $\varphi$ with degree at most $m(b+1)$, and
    - has $|\varphi|$ states, index $|\varphi|$. 
Solving the satisfiability problem

- We show that the satisfiability problem for enriched $\mu$-calculus formulas (except for fully enriched ones) is EXPTime-Complete.

- Lower Bound: Satisfiability for the $\mu$-calculus is EXPTime-hard [Fisher Ladner 1979].

- Upper Bound: We use an automata-theoretic approach:
  - Given a sentence $\phi$ of the full graded $\mu$-calculus that has $m$ at least sub-sentences and counts up to $b$, we can construct a FEA $A_\phi$ that
    - accepts the set of tree models of $\phi$ with degree at most $m(b+1)$, and
    - has $|\phi|$ states, index $|\phi|$.
  - Given a sentence $\phi$ of the hybrid graded/full $\mu$-calculus with $m$ at least subsentences, $k$ nominals, and counts up to $b$, we can build a FEA $A_\phi$ that
    - accepts all quasi forest models of $\phi$ with degree max{$k+1$, $m(b+1)$}, and
    - has $O(|\phi|^2)$ states, index $|\phi|$. 
Solving the satisfiability problem

- We show that the satisfiability problem for enriched \( \mu \)-calculus formulas (except for fully enriched ones) is \( \text{EXPTime-Complete} \)

- Lower Bound: Satisfiability for the \( \mu \)-calculus is \( \text{EXPTime-hard} \) [Fisher Ladner 1979]

- Upper Bound: We use an automata-theoretic approach:
  - Given a sentence \( \phi \) of the full graded \( \mu \)-calculus that has \( m \) at least sub-sentences and counts up to \( b \), we can construct a FEA \( A_\phi \) that
    - accepts the set of tree models of \( \phi \) with degree at most \( m(b+1) \), and
    - has \( |\phi| \) states, index \( |\phi| \).
  - Given a sentence \( \phi \) of the hybrid graded/full \( \mu \)-calculus with \( m \) at least subsentences, \( k \) nominals, and counts up to \( b \), we can built a FEA \( A_\phi \) that
    - accepts all quasi forest models of \( \phi \) with degree \( \max\{k+1, m(b+1)\} \), and
    - has \( O(|\phi|^2) \) states, index \( |\phi| \).

- In both cases, \( \phi \) is satisfiable if \( L(A_\phi) \neq \emptyset \)
Solving the emptiness problem

- We first reduce the emptiness problem for FEA to the emptiness problem for 2GAPTs.
  - A 2GAPT is a FEA that accepts trees and cannot jump to the root of the input tree.
Solving the emptiness problem

- We first reduce the emptiness problem for FEA to the emptiness problem for 2GAPTs.
  - A 2GAPT is a FEA that accepts trees and cannot jump to the root of the input tree.

- To decide the emptiness of 2GAPTs, we use a reduction to the emptiness problem of GNPT, via “strategy trees”
  - To remove alternation, we build special trees that allow encoding the original run in one having the same tree structure as the input tree.
  - To restrict to unidirectional paths, we use the notion of annotation that allow to decompose each path into downward paths and detours.
Solving the emptiness problem

- We first reduce the emptiness problem for FEA to the emptiness problem for 2GAPTs.
  - A 2GAPT is a FEA that accepts trees and cannot jump to the root of the input tree.

- To decide the emptiness of 2GAPTs, we use a reduction to the emptiness problem of GNPT, via “strategy trees”
  - To remove alternation, we build special trees that allow encoding the original run in one having the same tree structure as the input tree.
  - To restrict to unidirectional paths, we use the notion of annotation that allow to decompose each path into downward paths and detours.

- The result follows from the blow-up involved in building the GNPT and from the complexity for checking its emptiness.
A strategy tree with detour

Figure 2: A fragment of an input tree, a corresponding run, and its strategy tree.
A strategy tree with detour

Figure 2: A fragment of an input tree, a corresponding run, and its strategy tree.
A strategy tree with detour

Figure 2: A fragment of an input tree, a corresponding run, and its strategy tree.
A strategy tree with detour

Figure 2: A fragment of an input tree, a corresponding run, and its strategy tree.
A strategy tree with detour

Figure 2: A fragment of an input tree, a corresponding run, and its strategy tree.
A strategy tree with detour

Figure 2: A fragment of an input tree, a corresponding run, and its strategy tree.
**A Summary for Enriched $\mu$-calculi**

<table>
<thead>
<tr>
<th>Results on the satisfiability problem for Enriched $\mu$-calculi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inverse programs</td>
</tr>
<tr>
<td>fully enriched</td>
</tr>
<tr>
<td>full hybrid</td>
</tr>
<tr>
<td>full graded</td>
</tr>
<tr>
<td>hybrid graded</td>
</tr>
<tr>
<td>graded</td>
</tr>
<tr>
<td>full</td>
</tr>
</tbody>
</table>

1. [Bonatti, Peron 2004]
2. [Sattler, Vardi 2001]
3. [Vardi 1998]
4. [Calvanese, De Giacomo, Lenzerini, 2001]
5. [Kupferman, Sattler, Vardi, 2002]
A Summary for Enriched $\mu$-calculi

<table>
<thead>
<tr>
<th>Results on the satisfiability problem for Enriched $\mu$-calculi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inverse programs</td>
</tr>
<tr>
<td>-------------------</td>
</tr>
<tr>
<td>fully enriched</td>
</tr>
<tr>
<td>full hybrid</td>
</tr>
<tr>
<td>full graded</td>
</tr>
<tr>
<td>hybrid graded</td>
</tr>
<tr>
<td>graded</td>
</tr>
<tr>
<td>full</td>
</tr>
</tbody>
</table>

1. [Bonatti, Peron 2004]
2. [Sattler, Vardi 2001]
3. [Vardi 1998]
4. [Calvanese, De Giacomo, Lenzerini, 2001]
5. [Kupferman, Sattler, Vardi, 2002]
A Summary for Enriched $\mu$-calculi

<table>
<thead>
<tr>
<th>Results on the satisfiability problem for Enriched $\mu$-calculi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inverse programs</td>
</tr>
<tr>
<td>------------------</td>
</tr>
<tr>
<td>fully enriched</td>
</tr>
<tr>
<td>full hybrid</td>
</tr>
<tr>
<td>full graded</td>
</tr>
<tr>
<td>hybrid graded</td>
</tr>
<tr>
<td>graded</td>
</tr>
<tr>
<td>full</td>
</tr>
</tbody>
</table>

1. [Bonatti, Peron 2004]
2. [Sattler, Vardi 2001]
3. [Vardi 1998]
4. [Calvanese, De Giacomo, Lenzerini, 2001]
5. [Kupferman, Sattler, Vardi, 2002]
Enriching Temporal Logics

- μ-calculus is a very expressive but too low-level logic.
- Branching time temporal logics such as CTL, and CTL* are less expressive but much more human-friendly.
Enriching Temporal Logics

- μ-calculus is a very expressive but too low-level logic.
- Branching time temporal logics such as CTL, and CTL* are less expressive but much more human-friendly.
- What about enriching CTL and CTL* with graded modalities.
  - So far, only CTL has been fully solved, both in unary and binary coding.
  - Graded CTL is exponentially more succinct than graded μ-calculus.
  - The satisfiability problem remains ExpTime-Complete
Enriching Temporal Logics

- μ-calculus is a very expressive but too low-level logic.
- Branching time temporal logics such as CTL, and CTL* are less expressive but much more human-friendly.
- What about enriching CTL and CTL* with graded modalities.
  - So far, only CTL has been fully solved, both in unary and binary coding.
  - Graded CTL is exponentially more succinct than graded μ-calculus.
  - The satisfiability problem remains ExpTime-Complete
- Moving from μ-calculus to CTL with graded modalities, we need to move from graded world modalities to graded path modalities!
Syntax of GCTL* and GCTL

- GCTL* extends CTL* with new graded path quantifiers:
  - "there exists at least n paths satisfying a given property";
  - "all but at most n paths satisfy a given property".
Syntax of GCTL* and GCTL

- **GCTL*** extends CTL* with new graded path quantifiers:
  - "there exists at least n paths satisfying a given property";
  - "all but at most n paths satisfy a given property".

- **CTL*** uses state and path formulas built inductively as follows:

  - **State-formulas:**
    - $\varphi ::= p | \neg \varphi | \varphi \land \varphi | \varphi \lor \varphi | E^{\geq n} \psi | A^{< n} \psi$
    - where $p \in AP$ and $\psi$ is a path-formula

  - **path-formulas (LTL):**
    - $\psi ::= \varphi | \psi \land \psi | \neg \psi | X\psi | \psi U \psi$
    - where $\varphi$ is a state-formula, and $\psi$ a path-formula

- **GCTL** formulas are obtained by forcing each temporal operator to be coupled with a path quantifier.
What does counting paths mean?

- A property ensured by a common prefix may be satisfied on an infinite number of paths.
What does counting paths mean?
- A property ensured by a common prefix may be satisfied on an infinite number of paths.
- It may happen that the prefix satisfies a formula but a whole path may not.
Counting paths

- What does counting paths mean?
  - A property ensured by a common prefix may be satisfied on an infinite number of paths.
  - It may happen that the prefix satisfies a formula but a whole path may not.

- We restrict to minimal and conservative paths
- Two paths are equivalent if
  - their common prefix satisfy the formula.
  - no matter how these prefixes are extended in the structure, the paths satisfy the formula.
Semantics of GCTL*

- For a Kripke structure $K$, a world $w$, and a GCTL* path formula $\psi$,
- Let $P(K, w, \psi)$ be the set of minimal and conservative paths of $K$ starting in $w$ and satisfying $\psi$
Semantics of GCTL*

- For a Kripke structure K, a world w, and a GCTL* path formula ψ,
- Let P(K, w, ψ) be the set of minimal and conservative paths of K starting in w and satisfying ψ
  - \( K, w \models E^n \psi \) iff \(|P(K, w, \psi)| \geq n\)
  - \( K, w \models A^\leq n \psi \) iff \(|P(K, w, \neg \psi)| < n\)
- For n=1, we write Eψ and Aψ instead of E^1 ψ e A^1 ψ
Solving GCTL in unary coding

- Let $\psi$ be a GCTL formula with grades coded in unary.
- From $\psi$ we build in linear time a "Partitioning Alternating Büchi Tree Automata" (PABT) $P_\psi$. 
Solving GCTL in unary coding

- Let $\psi$ be a GCTL formula with grades coded in unary.
- From $\psi$, we build in linear time a “Partitioning Alternating Büchi Tree Automata” (PABT) $P_\psi$.
- A PABT accepts all tree models of a formula, by “guessing” how to partition a required graded modality among successors.
Solving GCTL in unary coding

- Let $\psi$ be a GCTL formula with grades coded in unary.
- From $\psi$ we build in linear time a “Partitioning Alternating Büchi Tree Automata” (PABT) $P_\psi$.
- A PABT accepts all tree models of a formula, by «gessing» how to partition a required graded modality among successors.

\[ E_3^3 F_\psi \]

\[ \begin{array}{c}
2 \\
F_\psi \\
1 \quad F_\psi 
\end{array} \]

- By means of an opportune extension of the Myhano-Hayashi technique, we translate in Exponential Time $P_\psi$ in an NBT $B_\psi$.
- Since the emptiness of $L(B_\psi)$ can be checked in polynomial time, we get that the satisfiability problem for GCTL is in ExpTime.
- ExpTime hardness comes from the satisfiability problem for CTL.
Solving GCTL in binary coding

- If we use the unary case approach, we lose an exponent:
  - The tree model property requires trees with a branching degree exponential in the highest graded $b_{\text{max}}$ of the formula.
Solving GCTL in binary coding

- If we use the unary case approach, we lose an exponent:
  - The tree model property requires trees with a branching degree exponential in the highest graded $b_{\text{max}}$ of the formula.

- We use a binary encoding of each tree model and split the automata construction into a linear PABT plus a satellite NBT automaton.
  - The tree encoding turns each level of the tree in a binary tree, i.e., brothers of a node become its successors.
  - The satellite is an (exponential) NBT and ensures that each tree model satisfies some structural properties along its paths.
Solving GCTL in binary coding

- If we use the unary case approach, we lose an exponent:
  - The tree model property requires trees with a branching degree exponential in the highest graded $b_{\text{max}}$ of the formula.

- We use a binary encoding of each tree model and split the automata construction into a linear PABT plus a satellite NBT automaton.
  - The tree encoding turns each level of the tree in a binary tree, i.e., brothers of a node become its successors.
  - The satellite is an (exponential) NBT and ensures that each tree model satisfies some structural properties along its paths.

- As the satellite automaton is already an NBT, this avoids to inject an extra exponent when moving both automata to a unique NBT.

- Thus, also in the binary coding, the satisfiability question for GCTL is ExpTime-complete.
What about GCTL*

- Solving graded CTL* is even more appealing.
- There are several questions to investigate.
- Is GCTL* more succinct than Graded μ-calculus?
- What about the satisfiability?
  - Using a slight variation of the previous reasoning used for GCTL, we get a 3ExpTime upper bound.
  - As CTL* satisfiability is 2ExpTime-complete, it is an open question to decide the exact complexity of the problem for GCTL*
Further directions about GCTL and GCTL*

- What about GCTL/ GCTL* plus backwards modalities?
- CTL and CTL* have been investigated with respect to (linear and branching) Past modalities.
- PCTL (PCTL*) is (2)ExpTime-complete.
- What about GCTL/GCTL over more enriched structures: Hierachical, pushown, weighted etc...
Enriched modalities vs. open systems

- Enriched mu-calculi has been investigated in the setting of module checking.

- Same results as in the satisfiability case:
  - Undecidable if we consider the fully enriched mu-calculus.
  - ExpTime-complete for every fragment.
## Conclusion

### Results on the satisfiability problem for Enriched \( \mu \)-calculi

<table>
<thead>
<tr>
<th></th>
<th>Inverse programs</th>
<th>Graded modalities</th>
<th>Nominals</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>fully enriched</td>
<td>( \times )</td>
<td>( \times )</td>
<td>( \times )</td>
<td>Undecidable[1]</td>
</tr>
<tr>
<td>full hybrid</td>
<td>( \times )</td>
<td></td>
<td>( \times )</td>
<td>ExpTime[2]</td>
</tr>
<tr>
<td>full graded</td>
<td>( \times )</td>
<td>( \times )</td>
<td></td>
<td>ExpTime 2ary (1ary[4])</td>
</tr>
<tr>
<td>hybrid graded</td>
<td>( \times )</td>
<td></td>
<td>( \times )</td>
<td>ExpTime 1ary/2ary</td>
</tr>
<tr>
<td>graded</td>
<td>( \times )</td>
<td></td>
<td></td>
<td>ExpTime 1ary/2ary[3]</td>
</tr>
<tr>
<td>full</td>
<td>( \times )</td>
<td></td>
<td></td>
<td>ExpTime[5]</td>
</tr>
</tbody>
</table>

### Results on the satisfiability problem for GCTL

<table>
<thead>
<tr>
<th>GCTL</th>
<th>Graded modalities</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( \times )</td>
<td>ExpTime 1ary/2ary</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GCTL</th>
<th>Graded modalities</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Past CTL</td>
<td>( \times )</td>
<td>ExpTime[6]</td>
</tr>
</tbody>
</table>

1. [Bonatti, Peron 2004]
2. [Sattler, Vardi 2001]
3. [Vardi 1998]
4. [Calvanese, De Giacomo, Lenzerini, 2001]
5. [Kupferman, Sattler, Vardi, 2002]
6. [Kupferman, Pnueli 1995]
References

  - Invited extended version of ICALP’06
  - Extended version of LICS’09 and CSL’10
  - Invited extended version of FOSSACS ’07 and LPAR’07

Thank you for your attention!